ASSIGNMENT-2

Submitted by: Aryan Raj, Roll No.- 170123010

Link for Traces: https://drive.google.com/open?id=1eVsHgUTlbmbUwRCDNsu1P0FKwquTM6Ly

Question 1:

Protocol	Percent Packets	Packets	Percent Bytes	Bytes	Bits/s	End Packets	End Bytes	End Bits/s
Frame	100.0	1814	100.0	736238	15 k	0	0	0
Ethernet	100.0	1814	3.4	25396	517	0	0	0
Internet Protocol Version 4	100.0	1814	4.9	36280	739	0	0	0
User Datagram Protocol	0.6	10	0.0	80	1	0	0	0
Data	0.6	10	0.0	360	7	10	360	7
Transmission Control Protocol	99.4	1804	91.2	671410	13 k	842	42491	866
Transport Layer Security	52.3	948	87.0	640359	13 k	946	632309	12 k
Data	0.9	16	0.1	964	19	16	964	19

Various protocols used by the application TeamViewer while using it are

1) Transport Layer: TCP and UDP 2) Network Layer: IPv4 3) Secure Socket/Session Layer: TLSv1.2 Protocol 4) Link Layer: Ethernet Protocol.

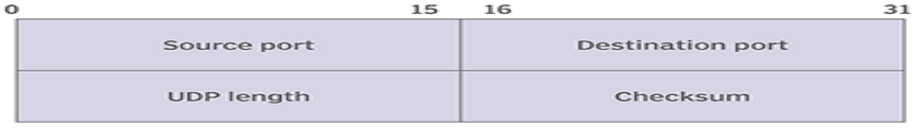
TLS (Transport layer Security):

- TLS protocol sits between the Application Layer and the Transport Layer.
- It aims primarily to provide privacy and data integrity between two communicating computer applications.
- The connection is private because symmetric cryptography is used to encrypt the data transmitted.
- The keys for this symmetric encryption are generated uniquely for each connection and are based on a shared secret negotiated at the start of the session (TLS Handshake).
- The connection ensures integrity because each message transmitted includes a message integrity check using a message authentication code to prevent undetected loss or alteration of the data during transmission.
- The TLS header comprises three fields, necessary to allow the higher layer to be built upon it:
  (i) Byte 0: TLS record type (example : CHANGE_CIPHER_SPEC , ALERT, HANDSHAKE,APPLICATION DATA)
  (ii) Bytes 1-2: TLS version (major/minor) (example : TLSv1.0, TLSv1.1, TLSv1.2)
  (iii) Bytes 3-4: Length of data in the record (excluding the header itself). The maximum supported is 16384 (16K).
- The TLS record contains the TLS protocol for example: Handshake protocol or change cipher spec protocol.

UDP (User Datagram Protocol):

UDP (User Datagram Protocol) is an alternative communications protocol to Transmission Control Protocol (TCP) used primarily for establishing low-latency and loss-tolerating connections between applications on the internet. UDP enables process-to-process communication. UDP sends messages, called datagrams, and is considered a best-effort mode of communications. In addition, where TCP provides error and flow control, no such mechanisms are supported in UDP. UDP is considered a connectionless protocol because it doesn't require a virtual circuit to be established before any data transfer occurs. The User Datagram Protocol header has four fields, each of which is 2 bytes. They are:

- source port number, which is the number of the sender;
- destination port number, the port the datagram is addressed to;
- length, the length in bytes of the UDP header and any encapsulated data; and
- checksum, which is used in error checking. Its use is required in IPv6 and optional in IPv4.



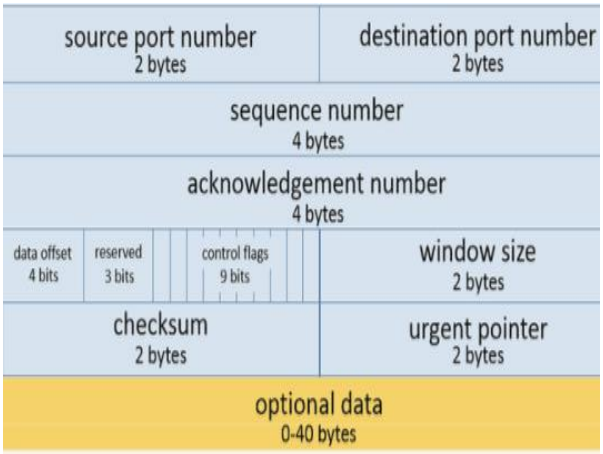
TCP (Transmission Control Protocol): TCP provides reliable, ordered, and error-checked delivery of a stream of octets (bytes) between applications running on hosts communicating by an IP network. A TCP packet is a sequence of octets (bytes) and consists of a header followed by a body. The body contains the data from the layer above it. The header describes the following:

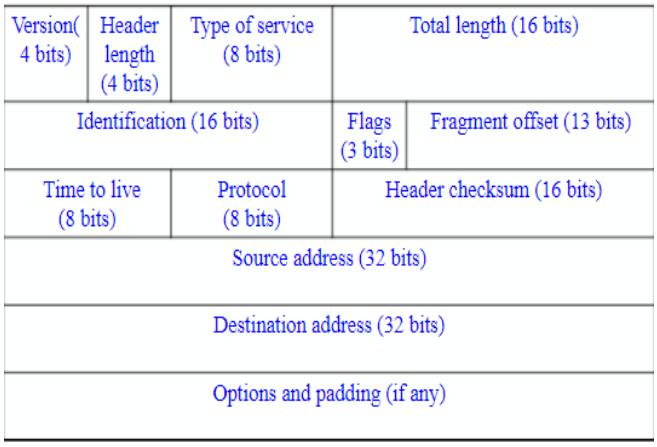
- Source and Destination Port(2 bytes each) : communication endpoints for sending and receiving devices.
- Sequence Number and Acknowledgment Number(4 bytes each) : Sequence numbers of the packets.
- TCP data offset (4 bits) : the total size of a TCP header in multiples of four bytes.
- Reserved data (3 bits) : his field serves the purpose of aligning the total header size as a multiple of four bytes
- Control flags (up to 9 bits): to manage data flow in specific situations.
- Window size (2 bytes): to regulate how much data they send to a receiver before requiring an acknowledgment in return
- TCP checksum (2 bytes): Error Control
- Urgent pointer (2 bytes): to regulate how much data they send to a receiver before requiring an acknowledgment in return
- TCP optional data (0-40 bytes): to regulate how much data they send to a receiver before requiring an acknowledgment in return

IPv4(Internet Protocol): IPv4 is a connectionless protocol for use on packet-switched networks. It operates on a best effort delivery model. The encapsulated data is referred to as IP Payload. IP header contains all the necessary information to deliver the packet at the other end.

- Version (4 bits): four-bit version field. For IPv4, this is always equal to 4.
- Internet Header Length (IHL)(4 bits): this field specifies the size of the header in multiples of 32 bits .The minimum value for this field is 5 which indicates a length of 5 × 32 bits = 160 bits = 20 bytes. As a 4-bit field, the maximum value is 15 words (15 × 32 bits, or 480 bits = 60 bytes).
- Differentiated Services Code Point (DSCP)(6 bits): This field is now defined for Differentiated services. New technologies are emerging that require real-time data streaming and therefore make use of the DSCP field. An example is Voice over IP (VoIP), which is used for interactive data voice exchange.
- Explicit Congestion Notification (ECN)(2 bits): allows end-to-end notification of network congestion without dropping packets.
- Total Length(16 bits): This 16-bit field defines the entire packet size in bytes, including header and data. The minimum size is 20 bytes (header without data) and the maximum is 65,535 bytes.
- Identification(16 bits): for uniquely identifying the group of fragments of a single IP datagram.
- Flags (3 bits): A three-bit field follows and is used to control or identify fragments. They are (i) bit 0: Reserved; must be zero, (ii) bit 1: Don't Fragment (DF) and (iii) bit 2: More Fragments (MF).
- Fragment Offset(13bits): specifies the offset of a particular fragment relative to the beginning of the original unfragmented IP datagram. The first fragment has an offset of zero. This allows a maximum offset of (213 – 1) × 8 = 65,528 bytes, which would exceed the maximum IP packet length of 65,535 bytes with the header length included (65,528 + 20 = 65,548 bytes).
- Time To Live (TTL)(8 bits): An eight-bit time to live field helps prevent datagrams from persisting the field has become a hop count— when the datagram arrives at a router, the router decrements the TTL field by one.
- Protocol(8 bits): This field defines the protocol used in the data portion of the IP datagram.
- Header Checksum(16 bits): The 16-bit checksum field is used for error-checking of the header.
- Source address(32 bits): This field is the IPv4 address of the sender of the packet.
- Destination address(32 bits): This field is the IPv4 address of the receiver of the packet.

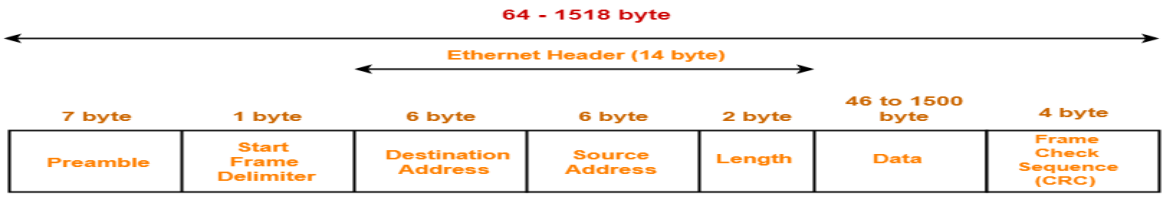
1&2) Figures on the left side, right side indicates frame format of TCP and IPv4 respectively.





<u>Ethernet (Data Link layer):</u> Data link layer protocols are point to point protocols. The data link layer is concerned with local delivery of frames between devices on the same LAN. Ethernet Frame Format:

- Preamble(7 bytes) : This is a stream of bits (alternating 1's and 0's) used to allow the transmitter and reciever to synchronize their communication.
- SFD(1 bytes) : This is always 10101011 and is used to indicate the beginning of the frame information.
- Destination/Source MAC(6 bytes each): This is the MAC address of the machine receiving/sending data. The MAC address is 6bytes longs. Initial 3 bytes are unique to each manufacturer. Last 3 bytes are unique to each hardware.
- Length (2 bytes): This is the length of the entire Ethernet frame in bytes.
- Data (Payload): The data is inserted here. This is where the data from IP layer is placed.
- FCS(4 bytes): This field contains the Frame Check Sequence (FCS) which is calculated using a Cyclic Redundancy Check (CRC). The FCS allows Ethernet to detect errors in the Ethernet frame and reject the frame if it appears damaged.



IEEE 802.3 Ethernet Frame Format

Question 2:

Observed values for various fields of the protocols:

```
> Frame 401: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface \Device\NPF_{82C227D4-B7EF-40E7-A3FF-8ED765321832}, id 0
> Ethernet II, Src: Cisco_97:1e:ef (4c:4e:35:97:1e:ef), Dst: Dell_f3:a5:51 (f4:8e:38:f3:a5:51)
v Internet Protocol Version 4, Src: 213.227.184.134, Dst: 10.3.3.61
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 40
    Identification: 0xc178 (49528)
  > Flags: 0x0000
    ...0 0000 0000 0000 = Fragment offset: 0
    Time to live: 62
    Protocol: TCP (6)
    Header checksum: 0x1fae [validation disabled]
    [Header checksum status: Unverified]
    Source: 213.227.184.134
    Destination: 10.3.3.61
> Transmission Control Protocol, Src Port: 443, Dst Port: 8735, Seq: 57562, Ack: 71547, Len: 0
```

1) **Frame 401**: The frame number of the trace I observed is 401. The frame protocol is not a real protocol itself, but used by Wireshark as a base for all the protocols on top of it. We also have information about the bytes used by the frame (60 bytes).

2) **Ethernet II:** Src: Cisco_97:1e:ef (4c:4e:35:97:1e:ef) : Hardware Mac address of my PC (Source). Dst: Dell_f3:a5:51 (f4:8e:38:f3:a5:51) : Hardware mac address of destination.

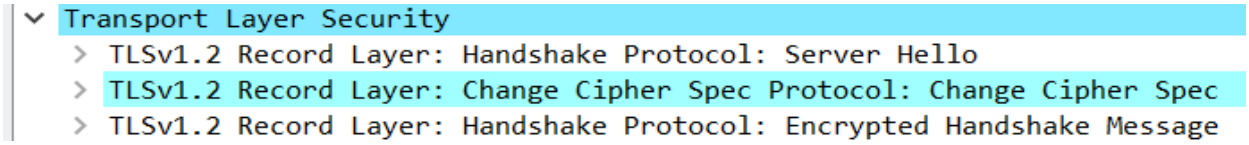
3) **Internet Protocol Version 4, Src: 213.227.184.134** - This destination computer IP address**, Dst: 10.3.3.61** – This is my computer IP address(Source IP Address). These IP addresses help us to uniquely identify both machines on the network. **Version: 4:** field indicates the format of the internet header. **Header length:** 20 bytes: Number of 32-bit words in the TCP header. **Differentiated Services Field (DSF): 0x00:** Indicates particular Quality of Service needs from the network, the DSF defines the way routers should queue packets while they are waiting to be forwarded. **Total Length: 40**: is the length of the datagram, measured in octets, including internet header and data. This field allows the length of a datagram to be up to 65,535 octets. **Flags:** 3 bits, Various Control Flags: (i) **Bit 0:** reserved, must be zero ,(ii) **Bit 1:** (DF) 0 = May Fragment, 1 = Don't Fragment and (iii) **Bit 2**: (MF) 0 = Last Fragment, 1 = More Fragments. **Fragment Offset: 0 bits**: This field indicates where in the datagram this fragment belongs. **Time to live: 62**: This field indicates the maximum time the datagram is allowed to remain in the internet system. Field called Protocol (Same as protocol number): **TCP (6):** To identify the next level protocol used**. Header checksum: 0x1fae:** A checksum on the header only and its status is unverified.

```
> Internet Protocol Version 4, Src: 213.227.184.134, Dst: 10.3.3.61
v Transmission Control Protocol, Src Port: 443, Dst Port: 8735, Seq: 57562, Ack: 71547, Len: 0
    Source Port: 443
    Destination Port: 8735
    [Stream index: 3]
    [TCP Segment Len: 0]
    Sequence number: 57562    (relative sequence number)
    Sequence number (raw): 1714048938
    [Next sequence number: 57562    (relative sequence number)]
    Acknowledgment number: 71547    (relative ack number)
    Acknowledgment number (raw): 1206743083
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window size value: 1213
    [Calculated window size: 155264]
    [Window size scaling factor: 128]
    Checksum: 0x7da6 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
```

4) **Transmission Control Protocol**:

- **Src Port**: 443: This is source port, which is sending packets.
- **Dst Port**: 8735: My computer port in this case which is receiving packets.
- **TCP Segment Length**: 0: This is TCP packet segment length.
- **Sequence number**:57562: If the SYN flag is set (1), then this is the initial sequence number. The sequence number of the actual first data byte and the acknowledged number in the corresponding ACK are then this sequence number plus 1.
- **Acknowledgment number**: 71547: The ACK flag is set to 1 which means the value of this field is the next sequence number that the sender is expecting.
- **Flags** (9 bits): It contains 9 1-bit flags. **ACK** (1 bit):1: indicates that the Acknowledgment field is significant. All packets after the initial SYN packet sent by the client should have this flag set. **PSH (1 bit):1: Push function** : Asks to push the buffered data to the receiving application. Remaining seven flags i.e. NS, CWR, ECE, URG, RST, SYN, FIN have value 0.
- **Window size value**: 128: This is the space for the incoming data.
- **Checksum** (16 bits):0x7da6: The 16-bit checksum field is used for error checking of the header and data.
- **Urgent pointer** (16 bits): 0: If the URG flag is set, then this 16-bit field is an offset from the sequence number indicating the last urgent data byte.

5) **TLSv1.2 Protocol:** Transport Layer Security (TLS): Some of the fields of the protocol can be seen. Some other messages includes Client Hello, Server Hello, Certificate, Server Key Exchange, Server Hello Done between client (213.227.184.134) and server (10.3.3.61) and TLS is implemented on two levels- The TLS Record protocol and the TLS Handshake protocol to ensure security, efficiency and extensibility.



Question 3:

Relevance of Protocols used by the application:

a) TCP: This was the majorly used protocol in while using TeamViewer to remotely control another computer. As TeamViewer is used to remotely control another computer, reliability is of utmost importance. Hence, the use of TCP is inevitable. It is a standard that defines how to establish and maintain a network conversation via which application programs can exchange data. Its responsibility includes end-to-end message transfer independent of the underlying network and structure of user data, along with error control, segmentation, flow control, and helps to minimize traffic congestion control. This handles all the packets relating to TCP 3-way connection and TLS handshaking protocol. It addresses numerous reliability issues in providing a reliable byte stream: data arrives in-order, data has minimal error (i.e., correctness), duplicate data is discarded and lost or discarded packets are resent. TCP is optimized for accurate delivery rather than timely delivery, as correct sequence of buffer to be fetched. TCP's bandwidth probing and congestion control will attempt to use all of the available bandwidth between server and client.

b) TLSv1.2: TLS was designed to operate on top of a reliable transport protocol such as TCP. Web servers use cookies to identify the web user. They are small piece of data stored into the web user's disk. TLS is used to protect session cookies on the rest of the sites from being intercepted to protect user accounts. The TLS protocol aims primarily to provide privacy and data integrity between two communicating computer applications. SSL and TLS are both cryptographic protocols utilizing X.509 certificates, public/private key encryption that provide authentication and data encryption between servers, machines and applications operating over a network. TLS provides verification of identity of server, which is as important as encryption. The goals of the TLS protocol are cryptographic security, extensibility, and relative efficiency. These goals are achieved through implementation of the TLS protocol on two levels: the TLS Record protocol and the TLS Handshake protocol. TLS allows the peers to negotiate a shared secret key without having to establish any prior knowledge of each other, and to do so over an unencrypted channel, Client-server applications use the TLS protocol to communicate across a network in a way designed to prevent eavesdropping and tampering. Therefore, TLS is used to prevent hackers to snoop on the packets being sent between the two endpoints.

c) TCP is compatible only with IP at Network Layer, so TeamViewer uses IP at the network layer (IPv4). The IP has the job of delivering packets using the IP headers from the source to the destination. Existing in the network layer, IPv4 connection is a hop to hop connection. It is a connectionless protocol, hence neither reliable delivery nor correctly ordered data is guaranteed. These aspects are taken care of by TCP.

d) Ethernet II is used at link layer as it ensures reliable data transfer between two nodes and involves proper error handling and flow control mechanisms to minimize errors. Other features include CRC for error detection and preamble for synchronization.

Question 4:

a) **3-Way TCP Handshake Message sequence**: Before a client attempts to connect with a server, the server must first bind to and listen at a port to open it up for connections: this is called a passive open. Once the passive open is established, a client may initiate an active open. Establishing a normal TCP connection requires three separate steps: **1) SYN**: The client sending a SYN to the server performs the active open. The client sets the segment's sequence number to a random value 'A'. **2) SYN-ACK**: In response, the server replies with a SYN-ACK (acknowledging our SYN request). The acknowledgment

number is set to one more than the received sequence number i.e. A+1, and the sequence number that the server chooses for the packet is another random number, 'B'. **3) ACK**: Finally, the client sends an ACK back to the server. The sequence number is set to the received acknowledgement value i.e. A+1, and the acknowledgement number is set to one more than the received sequence number i.e. B+1.

21 6.533897	169.56.146.131	10.3.3.61	SSL	733 Continuation Data
22 6.574432	10.3.3.61	169.56.146.131	TCP	54 8525 → 443 [ACK] Seq=3956 Ack=4066 Win=1026 Len=0
23 6.879448	10.3.3.61	213.227.184.134	TCP	66 8735 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
24 6.881939	213.227.184.134	10.3.3.61	TCP	66 443 → 8735 [SYN, ACK] Seq=0 Ack=1 Win=18352 Len=0 MSS=9176 SACK_PERM=1 WS=128
25 6.882122	10.3.3.61	213.227.184.134	TCP	54 8735 → 443 [ACK] Seq=1 Ack=1 Win=262656 Len=0

b) TLS Handshaking Message Sequence observed:

16 38.036689	10.3.3.61	13.67.116.41	TLSv1.2	294 Client Hello
17 38.037478	13.67.116.41	10.3.3.61	TCP	60 443 → 5859 [ACK] Seq=1 Ack=241 Win=19456 Len=0
18 38.383955	13.67.116.41	10.3.3.61	TLSv1.2	1514 Server Hello
19 38.384116	10.3.3.61	13.67.116.41	TCP	54 5859 → 443 [ACK] Seq=241 Ack=1461 Win=262144 Len=0
20 38.384183	13.67.116.41	10.3.3.61	TCP	1514 443 → 5859 [ACK] Seq=1461 Ack=241 Win=19456 Len=1460 [TCP segment of a reassembled PDU]
21 38.384184	13.67.116.41	10.3.3.61	TCP	1514 443 → 5859 [ACK] Seq=2921 Ack=241 Win=19456 Len=1460 [TCP segment of a reassembled PDU]
22 38.384246	10.3.3.61	13.67.116.41	TCP	54 5859 → 443 [ACK] Seq=241 Ack=4381 Win=262144 Len=0
23 38.384506	13.67.116.41	10.3.3.61	TLSv1.2	1514 Certificate [TCP segment of a reassembled PDU]
24 38.384508	13.67.116.41	10.3.3.61	TLSv1.2	1257 Certificate Status, Server Key Exchange, Server Hello Done
25 38.384645	10.3.3.61	13.67.116.41	TCP	54 5859 → 443 [ACK] Seq=241 Ack=7044 Win=262144 Len=0
26 38.390747	10.3.3.61	13.67.116.41	TLSv1.2	147 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
27 38.391267	13.67.116.41	10.3.3.61	TCP	60 443 → 5859 [ACK] Seq=7044 Ack=334 Win=19456 Len=0
28 38.486714	13.67.116.41	10.3.3.61	TLSv1.2	105 Change Cipher Spec, Encrypted Handshake Message
29 38.486840	10.3.3.61	13.67.116.41	TCP	54 5859 → 443 [ACK] Seq=334 Ack=7095 Win=261888 Len=0
30 38.488442	10.3.3.61	13.67.116.41	TLSv1.2	543 Application Data
31 38.488816	10.3.3.61	13.67.116.41	TCP	1514 5859 → 443 [ACK] Seq=823 Ack=7095 Win=261888 Len=1460 [TCP segment of a reassembled PDU]
32 38.488823	10.3.3.61	13.67.116.41	TLSv1.2	195 Application Data
33 38.489194	13.67.116.41	10.3.3.61	TCP	60 443 → 5859 [ACK] Seq=7095 Ack=823 Win=20608 Len=0

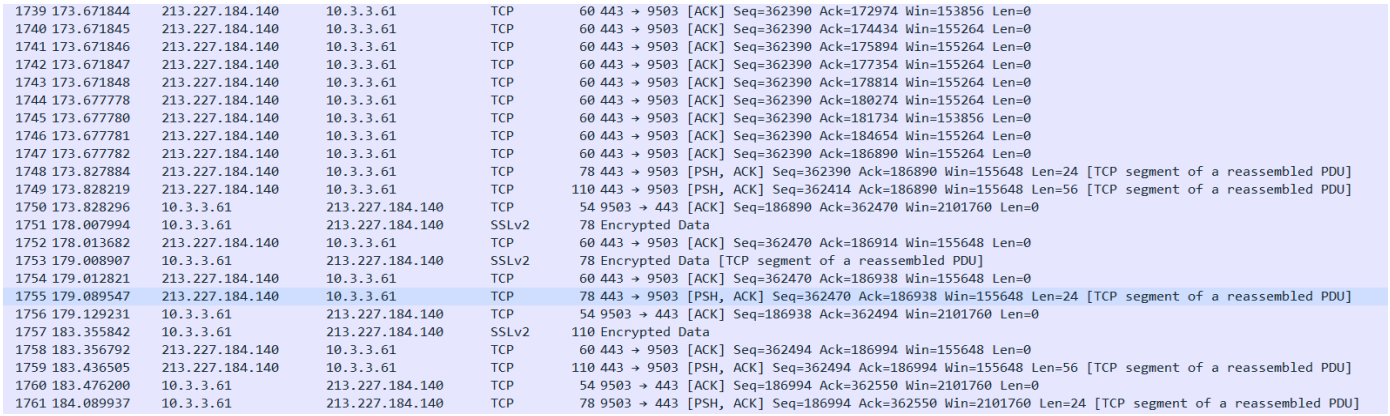
The record encapsulates a "control" protocol (the handshake messaging protocol) when the TLS connection starts. This protocol is used to exchange all the information required by both sides for the exchange of the actual application data by TLS and to negotiate the secure attributes of a session. It defines the messages formatting or containing this information and the order of their exchange. The usual steps are – (i) Client Hello: Client initiates a session with the server, along with sending version, session id, Cipher-suite, compression method etc. (ii) Server Response: Server responds back to client with its own Server Hello. It also sends the cipher key message and the encryption handshake message. The server finished message tells that TLS negotiation is completed on server side. (iii) Client Response: Client responds to server that it has understood the message and changes the record layer state to use symmetric key encryption and sends an encryption handshake message. Now, client is OK to send data.

The TLS Handshake protocol allows authenticated communication to commence between the server and client. As for TeamViewer it is not mentioned any particular function, so I chose to functionalities as establishing connection to control remote computer and play videos on youtube in the remote computer as two functions for carrying the observations.

(i) Establishing connection to control remote computer:

725 51.975114	10.3.3.61	216.58.197.74	TCP	66 9508 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
726 51.975709	216.58.197.74	10.3.3.61	TCP	66 443 → 9508 [SYN, ACK] Seq=0 Ack=1 Win=18352 Len=0 MSS=9176 SACK_PERM=1 WS=128
727 51.975798	10.3.3.61	216.58.197.74	TCP	54 9508 → 443 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
728 51.976138	10.3.3.61	216.58.197.74	TLSv1.3	651 Client Hello
729 51.976842	216.58.197.74	10.3.3.61	TCP	60 443 → 9508 [ACK] Seq=1 Ack=598 Win=19584 Len=0
730 51.978104	10.3.3.61	172.217.163.110	TCP	66 9509 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
731 51.978966	172.217.163.110	10.3.3.61	TCP	66 443 → 9509 [SYN, ACK] Seq=0 Ack=1 Win=18352 Len=0 MSS=9176 SACK_PERM=1 WS=128
732 51.979026	10.3.3.61	172.217.163.110	TCP	54 9509 → 443 [ACK] Seq=1 Ack=1 Win=262656 Len=0
733 51.979299	10.3.3.61	172.217.163.110	TLSv1.3	639 Client Hello
734 51.980108	172.217.163.110	10.3.3.61	TCP	60 443 → 9509 [ACK] Seq=1 Ack=586 Win=19584 Len=0
735 52.322278	216.58.197.74	10.3.3.61	TLSv1.3	266 Server Hello, Change Cipher Spec, Application Data
736 52.325190	10.3.3.61	216.58.197.74	TLSv1.3	118 Change Cipher Spec, Application Data
737 52.325671	10.3.3.61	216.58.197.74	TLSv1.3	140 Application Data
738 52.325688	216.58.197.74	10.3.3.61	TCP	60 443 → 9508 [ACK] Seq=213 Ack=662 Win=19584 Len=0
739 52.325733	172.217.163.110	10.3.3.61	TLSv1.3	266 Server Hello, Change Cipher Spec, Application Data
740 52.326319	216.58.197.74	10.3.3.61	TCP	60 443 → 9508 [ACK] Seq=213 Ack=748 Win=19584 Len=0
741 52.326806	10.3.3.61	216.58.197.74	TLSv1.3	874 Application Data
742 52.327465	216.58.197.74	10.3.3.61	TCP	60 443 → 9508 [ACK] Seq=213 Ack=1568 Win=21248 Len=0
743 52.332654	10.3.3.61	172.217.163.110	TLSv1.3	118 Change Cipher Spec, Application Data
744 52.333067	10.3.3.61	172.217.163.110	TLSv1.3	140 Application Data
745 52.333382	172.217.163.110	10.3.3.61	TCP	60 443 → 9509 [ACK] Seq=213 Ack=650 Win=19584 Len=0
746 52.333680	172.217.163.110	10.3.3.61	TCP	60 443 → 9509 [ACK] Seq=213 Ack=736 Win=19584 Len=0
747 52.333988	10.3.3.61	172.217.163.110	TLSv1.3	1364 Application Data

(ii) Playing videos on youtube:



```
1739 173.671844   213.227.184.140   10.3.3.61         TCP        60 443 → 9503 [ACK] Seq=362390 Ack=172974 Win=153856 Len=0
1740 173.671845   213.227.184.140   10.3.3.61         TCP        60 443 → 9503 [ACK] Seq=362390 Ack=174434 Win=155264 Len=0
1741 173.671846   213.227.184.140   10.3.3.61         TCP        60 443 → 9503 [ACK] Seq=362390 Ack=175894 Win=155264 Len=0
1742 173.671847   213.227.184.140   10.3.3.61         TCP        60 443 → 9503 [ACK] Seq=362390 Ack=177354 Win=155264 Len=0
1743 173.671848   213.227.184.140   10.3.3.61         TCP        60 443 → 9503 [ACK] Seq=362390 Ack=178814 Win=155264 Len=0
1744 173.677778   213.227.184.140   10.3.3.61         TCP        60 443 → 9503 [ACK] Seq=362390 Ack=180274 Win=155264 Len=0
1745 173.677780   213.227.184.140   10.3.3.61         TCP        60 443 → 9503 [ACK] Seq=362390 Ack=181734 Win=153856 Len=0
1746 173.677781   213.227.184.140   10.3.3.61         TCP        60 443 → 9503 [ACK] Seq=362390 Ack=184654 Win=155264 Len=0
1747 173.677782   213.227.184.140   10.3.3.61         TCP        60 443 → 9503 [ACK] Seq=362390 Ack=186890 Win=155264 Len=0
1748 173.827884   213.227.184.140   10.3.3.61         TCP        78 443 → 9503 [PSH, ACK] Seq=362390 Ack=186890 Win=155648 Len=24 [TCP segment of a reassembled PDU]
1749 173.828219   213.227.184.140   10.3.3.61         TCP       110 443 → 9503 [PSH, ACK] Seq=362414 Ack=186890 Win=155648 Len=56 [TCP segment of a reassembled PDU]
1750 173.828296   10.3.3.61         213.227.184.140   TCP        54 9503 → 443 [ACK] Seq=186890 Ack=362470 Win=2101760 Len=0
1751 178.007994   10.3.3.61         213.227.184.140   SSLv2      78 Encrypted Data
1752 178.013682   213.227.184.140   10.3.3.61         TCP        60 443 → 9503 [ACK] Seq=362470 Ack=186914 Win=155648 Len=0
1753 179.008907   10.3.3.61         213.227.184.140   SSLv2      78 Encrypted Data [TCP segment of a reassembled PDU]
1754 179.012821   213.227.184.140   10.3.3.61         TCP        60 443 → 9503 [ACK] Seq=362470 Ack=186938 Win=155648 Len=0
1755 179.089547   213.227.184.140   10.3.3.61         TCP        78 443 → 9503 [PSH, ACK] Seq=362470 Ack=186938 Win=155648 Len=24 [TCP segment of a reassembled PDU]
1756 179.129231   10.3.3.61         213.227.184.140   TCP        54 9503 → 443 [ACK] Seq=186938 Ack=362494 Win=2101760 Len=0
1757 183.355842   10.3.3.61         213.227.184.140   SSLv2     110 Encrypted Data
1758 183.356792   213.227.184.140   10.3.3.61         TCP        60 443 → 9503 [ACK] Seq=362494 Ack=186994 Win=155648 Len=0
1759 183.436505   213.227.184.140   10.3.3.61         TCP       110 443 → 9503 [PSH, ACK] Seq=362494 Ack=186994 Win=155648 Len=56 [TCP segment of a reassembled PDU]
1760 183.476200   10.3.3.61         213.227.184.140   TCP        54 9503 → 443 [ACK] Seq=186994 Ack=362550 Win=2101760 Len=0
1761 184.089937   10.3.3.61         213.227.184.140   TCP        78 9503 → 443 [PSH, ACK] Seq=186994 Ack=362550 Win=2101760 Len=24 [TCP segment of a reassembled PDU]
```

Question 5:

Time of the day (in 12 hrs format)	Throughput (Packets per sec)	RTT(ms)	Avg Packet Size (bytes)	Packets Lost	UDP Packets	TCP packets	Response per request
07:00 AM	3.3	38.1	337	0	10	1252	1.13
03:00 PM	4.8	39.2	401	0	97	2250	1.14
10:00 PM	4.1	41.2	379	0	16	2958	1.22

Question 6:

We can see that most of the IP addresses are same during different times of the day. Some IPs are different because when you run TeamViewer, you are assigned an ID on their broker server. You make a connection to a Teamviewer ID, and TeamViewer passes the connection down through the TeamViewer client's established tunnel to the destination and you then you are prompted for password and then the connection establishes afterwards. As TeamViewer client's established tunnel may change, so, IP may change.

Time of the day (in 12 hrs format)	Source IPs recorded*
07:00 AM	213.227.184.138,37.252.244.134
03:00 PM	213.227.184.134,216.58.197.78,169.56.146.131, 172.217.160.142, 172.217.163.131
10:00 PM	213.227.184.140, 213.227.184.138,172.217.160.142, 172.217.163.110

* not showing IPs for which the number of packets sent was very low (<10 packets)